Reconnaissance et génération de plan d'actions : application à la réalisation de systèmes auto-explicatifs

Alain Michard

INRIA

CENTRE
SOPHIA ANTIPOLIS

Institut National
de Recherche
en Informatique
et en Automatique

Domaine de Voluceau
Rocquencourt
B.P. 105
78153 Le Chesnay Cedex
France
Tél. (3) 954 90 20

(219) Rapports de Recherche

N° 382

RECONNAISSANCE ET GÉNÉRATION DE PLAN D'ACTIONS : APPLICATION À LA RÉALISATION DE SYSTÈMES AUTO-EXPLICATIFS

Alain MICHARD

Mars 1985

120

RECONNAISSANCE ET GENERATION DE PLAN D'ACTIONS: APPLICATION A LA REALISATION DE SYSTEMES AUTO-EXPLICATIFS

Alain MICHARD

I.N.R.I.A. Centre de Sophia Antipolis

RESUME

Le volume et la complexité de la documentation technique fournie avec un gros progiciel peuvent constituer un obstacle à son utilisation par un public parfois peu averti des subtilités de la technique informatique. Pour faciliter l'abord puis l'utilisation des systèmes interactifs, il est devenu courant de prévoir un sous-système d'information en ligne capable de fournir à la demande une description des fonctionnalités du système principal. Mais cette information fonctionnelle est souvent inadéquate, l'utilisateur souhaitant obtenir des conseils sur les procèdures à suivre pour réaliser une tâche donnée, autrement dit sur un plan d'actions capable de l'amener au resultat recherché. Les techniques developpées ces dernières années en Intelligence Artificielle dans le domaine de la reconnaissance et de la génération de plans permettent de créer des systèmes auto-explicatifs répondant à ce besoin. Après une introduction aux principes sous-jacents à ces méthodes, nous montrons comment nous les avons appliquées à la création d'un éditeur auto-explicatif et nous defendons l'idée d'une généralisation de leur usage à la plupart des futurs grands logiciels interactifs.

PLAN GENERATION AND PLAN RECOGNITION: APPLICATION TO THE DESIGN OF ADVICE-GIVING SYSTEMS

Alain MICHARD

I.N.R.I.A. Centre de Sophia-Antipolis

ABSTRACT

It is common for the new user of an interactive software to be somewhat discouraged by the volume and the complexity of the technical documentation he has to read and -hopefully- understand. A traditionnal way the system designer has to meet this difficulty is to offer an on-line help system to provide functionnal description -syntactic and semantic definition of each command-. This functionnal information is often inadequate, for the user wishes in fact advice on the exact procedure -the set of commands- that must be used to perform a given task. Plan recognition an plan generation techniques are of great interest to the design of advice-giving systems. The principles of this methodology are described and an illustration is given through the implementation of an advice-giving editor. Generalisation of this kind of assistance to most futur large interactive systems is advocated.

1. INTRODUCTION.

La plupart des grands logiciels interactifs actuels sont fournis avec un système de documentation en ligne, capable de fournir une assistance aux utilisateurs sous forme de descriptions des fonctionnalités disponibles. Ces descriptions contiennent en fait deux types au moins d'informations: syntaxe de la commande, et sémantique c'est à dire description des effets de celle-ci. La clef d'indexation de cette documentation est généralement constituée par les noms des commandes elles-mêmes. Des exemples typiques de ces logiciels de documentation, bien connus des informaticiens, sont le "help" du système Multics, ou le "man" de Unix.

Une possibilité supplémentaire est parfois proposée sous forme de recherche d'occurence d'un mot-clef quelconque dans tout ou partie du corpus documentaire. Pour les initiés, nous citerons en exemple la fonction "apropos" de l'éditeur emacs, ou le "list_help" de Multics.

La limitation commune à tous ces systèmes d'aide en ligne, est que l'on suppose toujours que l'utilisateur est capable de faire de bonnes hypothèses sur la suite des opérations à effectuer pour parvenir au résultat recherché (la suite des fonctions qu'il devra utiliser) et que la documentation en ligne lui sert à vérifier les conditions d'utilisation des commandes associées. Or, certains travaux issus de l'ergonomie du logiciel [8] tendent à montrer que la principale difficulté de l'utilisateur confronté à un logiciel interactif qu'il connaît mal, est de construire une suite d'actions qui le conduisent à son objectif, ou en d'autres termes de se construire une stratégie de résolution de sa tâche problème compatible avec les contraintes imposées par le système utilisé: avant même de chercher à s'informer sur la syntaxe et la sémantique précises d'une commande, il faut avoir idée de l'existence de cette commande et surtout de sa pertinence dans le contexte formé par la tâche en cours.

L'objectif commun à tous les logiciels explicatifs doit donc être d'aider l'utilisateur à planifier son action, c'est- à-dire de lui suggérer une suite de transformations convergeante vers l'objectif recherché. Etant donnée une tâche que l'utilisateur veut mener à bien, définie par un état initial du système et par un état final auquel il veut parvenir, l'explication consiste à lui suggérer un mode opératoire (i.e. une suite de transformations à appliquer au système) à un niveau de détail dépendant à la fois du type de tâche et du savoir disponible chez cet utilisateur. L'explication doit donc être fondée sur trois types de connaissances:

- la description fonctionnelle du système (ensemble des fonctions élémentaires qu'il peut réaliser),

- la description du ou des plans permettant de réaliser la tâche (un plan étant un arbre de buts et sous-buts dont les feuilles sont des transformations élémentaires). A chaque tâche et à chaque mode opératoire possible correspond un plan.

- une "représentation de l'utilisateur", c'est à dire des

indications sur l'ensemble des connaissances dont ce dernier dispose à un instant donné sur le fonctionnement du système.

Avant de préciser comment ces diverses connaissances peuvent être exploitées pour fournir une explication, précisons que le rôle d'un système explicatif n'est pas de permettre l'apprentissage ab initio d'un outil informatique. Les exemples existants [1, 14] et notre propre expérience, montrent que cette forme d'assistance est particulièrement bien adaptée à deux types d'utilisateurs : ceux qui n'ayant qu'un usage très occasionnel de cet outil oublient le détail des procédures de travail rarement employées, et ceux qui à partir de la connaissance d'un certain logiciel cherchent à découvrir le fonctionnement d'un nouvel outil de même type. Par exemple l'utilisateur d'un système de CAO donné ayant à utiliser un nouveau logiciel de CAO trouvera dans le système explicatif une aide précieuse pour l'aider à transférer son savoir-faire, mais il est sans doute illusoire de penser qu'un utilisateur néophyte puisse découvrir complètement ce qu'est la CAO uniquement grâce à un logiciel explicatif: pour poser de bonnes questions il doit déjà avoir une idée précise des tâches possibles, et posséder les notions de base propres au domaine d'application.

2. LE LOGICIEL EXPLICATIF PLANEX : OPTIONS FONDAMENTALES.

Deux grands types de systèmes explicatifs ont été décrits dans la littérature.
Dans le premier (voir [5] par exemple), le système est actif en permanence et tente de reconnaître dans la suite des actions de l'utilisateur un plan typique, c'est-à-dire d'inférer quel est l'objectif qu'il cherche à atteindre. Au cas où la suite des commandes émises diverge d'avec le plan "reconnu", le système explicatif intervient spontanément pour proposer une stratégie plus adaptée.
Dans le second mode [1, 14], le système n'est actif que sur demande de l'utilisateur: à tout instant au cours de son travail, celui-ci peut demander au système explicatif un mode opératoire permettant de parvenir à un certain résultat. La question est posée en langue naturelle plus ou moins restreinte. Le système explicatif exploite une représentation du système cible (système expliqué),une connaissance de l'état présent et une connaissance des effets des différentes commandes pour générer la suite de commandes (le plan) satisfaisante. Ce mode de fonctionnement n'est donc pas intrusif comme celui décrit plus haut, mais suppose qu'à partir de tout état courant possible (même résultant d'une suite de manipulations erronées) on sache proposer à l'utilisateur un plan le rapprochant de son objectif.

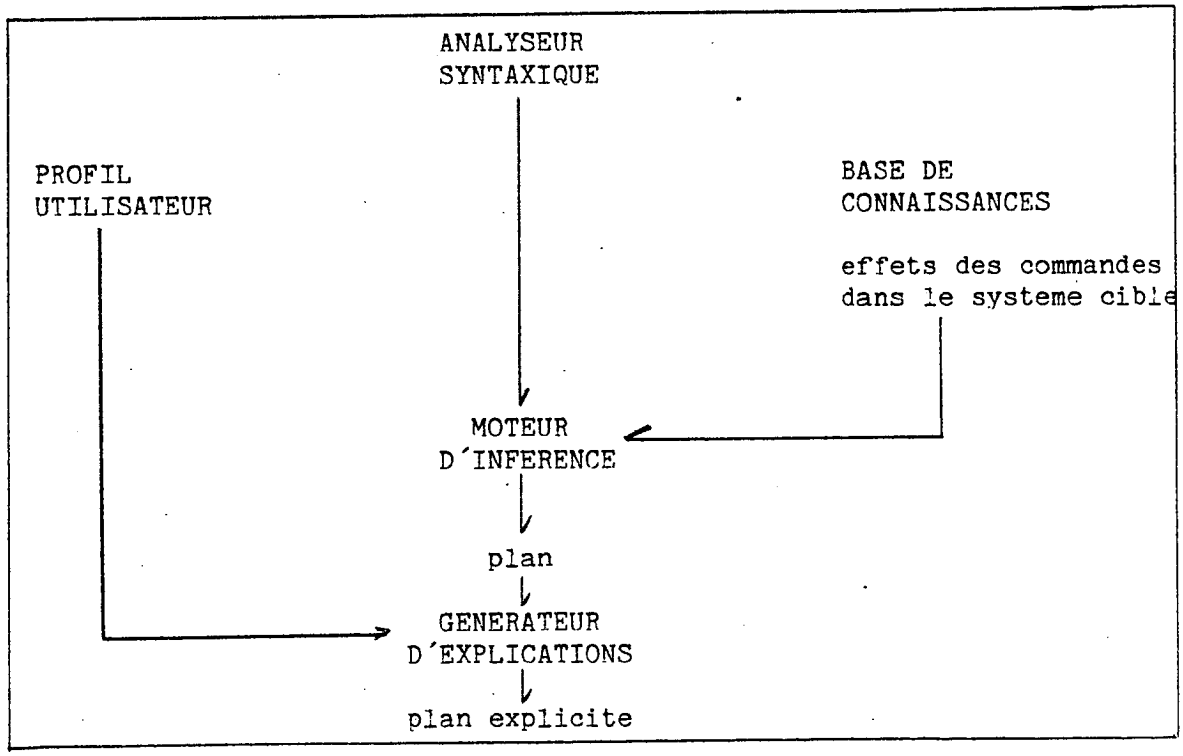
Nous avons choisi pour notre part la seconde approche en raison de deux critères, l'un d'ordre ergonomique, l'autre d'ordre technique. Le premier est qu'il nous paraît important pour faciliter l'apprentissage du système cible par l'utilisateur, de préserver son droit à l'erreur, et donc sa possibilité d'essayer des manipulations (commandes) parfois mal à propos, sans se voir rappeler à l'ordre par un "tuteur électronique". Dans le dialogue homme-machine, nous choisissons donc de privilégier l'initiative humaine. Le second critère est qu'il nous semble difficile de faire de la reconnaissance de plan, en se basant sur une

représentation du système cible suffisemment modulaire, aisément modifiable et extensible (l'approche choisie en [5] nous paraît révélatrice de cette difficulté), alors que nous verrons que la génération de plan peut se faire à partir d'une représentation à base de règles de production qui a de bonnes propriétés à cet égard.

3. ARCHITECTURE GENERALE DE PLANEX.

PLANEX est un ensemble de logiciels permettant de construire assez rapidement un système explicatif particulier. L'élément central est un générateur de plans indépendant d'un domaine donné, capable d'exploiter une base de connaissances spécifique à une "cible" pour générer une suite de transformations permettant de passer d'un état courant à un état souhaité. L'état souhaité (objectif de l'utilisateur) va être spécifié en langue naturelle et traduit en une représentation interne exploitable par le générateur de plans. Le plan généré sera lui-même traité, en particulier pour éviter de fournir des informations redondantes ou inutiles pour un utilisateur particulier, et présenté à l'utilisateur sous une forme facilement compréhensible.

La figure 1 présente les grandes unités fonctionnelles utilisées.



3.1. Analyse Syntaxique.

La liberté d'expression offerte à l'utilisateur dans la formulation de ses questions est très importante étant donné que les systèmes explicatifs sont principalement destinés à guider des débutants ou des non

spécialistes de l'application cible. Par ailleurs l'analyseur choisi doit être assez facilement adaptable à une application donnée utilisant un vocabulaire spécifique. Ces exigences nous ont conduit à utiliser le logiciel d'analyse du langage naturel ESOPE, développé à l'IRISA [10] qui possède les caractéristiques requises.

ESOPE, écrit en PROLOG permet de traduire toute phrase francaise formée à partir d'un vocabulaire fixé, en une représentation interne arbitraire, compatible avec l'application interfacée, ici le générateur de plans. Le lecteur intéressé trouvera en [7] des indications sur la méthode utilisée pour spécifier le vocabulaire et la grammaire du langage d'interrogation proposé à l'utilisateur, et sur la grammaire du langage interne défini dans le cadre d'une application (éditeur auto-explicatif).

3.2. Mécanisme de planification et représentation du système cible.

La génération de plans en intelligence artificielle obéit classiquement au paradigme suivant:

-Soit un état actuel du monde (dans notre cas un état du système cible) défini comme un ensemble E(0) de faits élémentaires vérifiés;

-on spécifie un objectif, état final souhaité, sous forme d'un ensemble de faits élémentaires E(f) qui ne sont pas tous vrais dans E(0);

- on dispose d'un ensemble de règles de transition qui associent un ensemble de prérequis (faits devant être vérifiés pour pouvoir appliquer la transition) et un ensemble de conséquences, l'une ou plusieurs de ces conséquences pouvant éventuellement appartenir à E(f);

-on définit enfin une procédure permettant de choisir une suite de transitions et une instanciation des variables impliquées dans ces transitions, telle que E(f) soit un sous-ensemble de l'ensemble des conséquences obtenues. Dans le schéma classique, cette procédure est une variante du mécanisme décrit sous le nom de "chaînage arrière" dans la littérature [13]: les prérequis de toute transition ayant un ou des éléments de E(f) en conclusion deviennent des buts intermédiaires que l'on recherche à leur tour dans les conclusions des règles de transition, et ainsi de suite jusqu'à retrouver E(0). La difficulté bien connue [9] consiste à trouver un cheminement valide tel qu'une transition choisie avec une instanciation de variables particulières n'ait pas une interaction négative avec une transition selectionnée antérieurement. Précisons tout de suite que le type de planification que nous visons offre très peu d'occasions de "blocages" tels que ceux décrits dans les mondes de blocs à déplacer : les conflits de ressources correspondent en principe à des opérations interdites ou impossibles, et nous proposons une méthode simple pour prendre en compte ces situations (règles de cohérence, cf infra).

Le choix fondamental qui doit être fait pour construire un générateur de plans est celui de la représentation que l'on va utiliser pour représenter l'état du monde d'une part et les transitions d'autre part. De nombreuses possibilités ont été décrites [9, 12, 15]. Nous avons choisi une représentation sous forme de règles de production exprimées dans un langage du type calcul des prédicats du premier ordre. Les raisons majeures de ce choix étant le bon savoir-faire disponible

avec cette méthodologie (1), et le fait que la notion de prototype ne soit pas indispensable pour décrire les fonctionnalités des systèmes interactifs qui constituent nos "micro-mondes" possibles.

3.3. Base de connaissances.

Le contenu de la base de connaissances (BC) est évidemment dépendant de l'application cible. Toutefois la BC sera toujours structurée de la même facon: on y trouvera principalement des règles définissant les opérateurs de transition, et les règles définissant les états. Les premières sont de la forme:

SI état I
ET opérateur T1
ALORS état J

Les opérateurs peuvent éventuellement être décomposés grace à des règles telles que:

SI état K
ET opérateur Ta
ET opérateur Tb
ALORS opérateur T1

SI état K
ET commande X
ALORS opérateur T2

Les commandes sont les feuilles du "plan-arbre" et ne sont pas décomposables. La factorisation des commandes en opérateurs et de ceux-ci en opérateurs plus généraux permet la création d'un plan hiérarchisé beaucoup plus proche de la représentation cognitive que l'utilisateur peut avoir de sa tâche, que ne le serait une suite non structurée de commandes. Cette hiérarchisation des opérateurs de transition facilite non seulement la génération d'une explication exploitable et compréhensible pour l'utilisateur final, mais aussi la constitution de la BC par l'expert.
Les états sont comme nous l'avons dit précedemment des ensembles de faits vrais à un instant donné, un état pouvant donc être inclus dans un autre. Les règles qui les définissent sont donc de la forme:

SI fait A
ET fait B
....
ALORS état I

La vérification de la valeur d'un fait (vrai/faux) se fait par des fonctions accédant à une structure de données servant d'interface entre le système cible et PLANEX. Cette structure de données doit refléter à tout instant l'état réel du système cible.

Outre ces règles qui permettent déja la planification, le système utilise des règles dites "de cohérence" qui permettent de déceler les

(1) Plusieurs des fonctions de bas niveau de PLANEX sont reprises du moteur d'inférence CRIQUET développé à l'INRIA [13]. Nous remercions ici P. Vignard de sa collaboration.

incohérences du type tentative d'utilisation d'un opérateur incompatible avec l'état courant. Ces incohérences peuvent être soit liées à une demande de l'utilisateur impossible à satisfaire, soit générées par le planificateur lui-même. Les règles de cohérence ont la forme:

SI état X
ET opérateur Tx
ALORS erreur N

et sont activées lors d'une recherche "en chaînage avant". L'activation d'une telle règle provoquera l'émission vers l'utilisateur d'un message si "opérateur Tx" a été directement demandé par celui-ci, et provoquera l'abandon de la branche en cours de recherche dans tous les cas.

3.4. Le plan généré et l'explication fournie.

L'objectif étant spécifié au générateur sous forme d'un état à satisfaire F, le résultat d'une recherche réussie aura la forme:

```
Etat 0
Opérateur T1
        Commande A
        Commande B
Opérateur T2
        Opérateur T21
                Commande D
                Commande E
        Opérateur T22
     .....

Etat F
```

Etat 0 correspondant à l'état courant du système cible.

Ce plan sous sa forme brute est le plus souvent beaucoup trop "riche" pour être présenté tel quel à l'utilisateur: celui-ci connaît fort bien le détail de certains opérateurs et ne souhaite pas se les voir rappeler dans tous les détails lors de chaque demande d'explication: il souhaite voir apparaître une représentation "holophrastée" de cet arbre, la profondeur visualisée dépendant à la fois de son degré d'expérience avec le système cible, et de la complexité (profondeur) de l'arbre complet. En particulier nous faisons l'hypothèse qu'un opérateur qui a déja été présenté "en détail" à un utilisateur donné, ne doit pas l'être de nouveau lors d'une explication ultérieure sauf demande explicite contraire.

Cette exigence nous amène à inclure dans le système une base de données, le "profil des utilisateurs" qui contient pour chaque utilisateur identifié du système la liste des opérateurs qui lui ont été décrit précedemment. La procédure de visualisation du plan consiste à présenter l'arbre jusqu'au niveau 2, en excluant les opérateurs figurant dans le profil de l'utilisateur. Par exemple, si pour le plan ci-dessus Opérateur T2 figure dans le profil de l'utilisateur concerné, le plan qui lui sera présenté aura la structure:

```
Etat 0
Opérateur T1
        Commande A
        Commande B
Opérateur T2
Opérateur T3
...
Etat f
```

Dans une version ultérieure de ce logiciel, il est prévu que ce plan présenté puisse être exploité comme un menu grâce à un dispositif de désignation (souris), le pointage de n'importe quel opérateur non terminal provoquant la visualisation du sous-arbre associé, et le pointage d'une commande (opérateur terminal) provoquant l'apparition d'une description fonctionnelle classique de la-dite commande. Dans la première version, la frappe littérale du nom d'un opérateur provoque l'expansion du sous-arbre correspondant et son affichage à l'écran.

4. UN EXEMPLE D'APPLICATION : UN EDITEUR VIDEO AUTO-EXPLICATIF.

A titre d'exemple, nous avons développé un système explicatif pour un éditeur plein-écran multi-tampons et mono-fenêtre (sous-ensemble de emacs). Ce choix étant justifié par la complexité de ce logiciel, suffisante pour que la BC ait une ampleur "réaliste" mais assez limitée pour que sa réalisation puisse être relativement rapide.

4.1. Base de connaissances.

L'état du système cible est représentée par quelques variables booléennes qui indiquent principalement les positions particulières du curseur (début ou fin de tampon, de paragraphe, de ligne, de mot), et l'état des différents tampons de travail (noms, noms des fichiers contenus, modification..).

Les règles décrivant les transitions obéissent au modèle général décrit précedemment. Exemples de règles (un peu simplifiées):

```
SI NON(tampon courant nil)
ET opérateur marquage ?quantite
ALORS marque ?quantite
```

```
SI marque ?quantite
ET commande wipe-region
ALORS effacement ?quantite
ET copie ?quantite anneau
```

dont la signification est

SI le tampon de travail courant n'est pas vide
ET si on applique l'opérateur de marquage sur un partie quelconque du texte
ALORS cette partie de texte est marquée.

SI une partie de texte est marquée
ET si on applique la commande "wipe-region"
ALORS on efface cette partie de texte
ET on la copie dans l'anneau d'effacement

6. BIBLIOGRAPHIE.

[1] Cullingford R.E., Krueger M.W., Selfridge M.,Bienkowski M.A.; Automated explanations as a component of a computer-aided design system. IEEE Transactions on Systems, Man and Cybernetics, 12,2,1982.

[2] Fenchel R.S., Estrin G.; Self-describing systems using integral help. IEEE Transactions on Systems, Man and Cybernetics, 12,2,1982.

[3] Chin D.N.; A case study in knowledge representation in UC. IJCAI 1983.

[4] Finin T.W.; Providing help and advice in task oriented systems. IJCAI 1983.

[5] Fischer G. Lemke A., Schwab T.; Active help systems. Readings on cognitive ergonomics. Lecture Notes in Computer Science. Springer Verlag. 1984.

[6] Jackson P., Lefrere P.; On the application of rule-based techniques to the design of advice-giving systems. Int. J. of Man-Machine Studies 1984, 20, 63-86.

[7] Michard A.; Comment utiliser PLANEX pour réaliser un système explicatif. Rapport technique INRIA, à paraitre.

[8] Richard J.F.; Logique du fonctionnement et logique de l'utilisation. Rapport de recherche INRIA No 202, Avril 1983.

[9] Sacerdoti E.; A structure for plans and behaviour. North Holland, New York, 1977.

[10] Saint Dizier P.; Traitement du langage naturel et programmation en logique. Note interne, IRISA, 1984.

[11] Schmidt C.F.; The role of object knowledge in human planning. Research Report CBM-TM-87 June 1980 Rutgers University.

[12] Sridharan N.S., Bresina J.L., Schmidt C.F.; Evolution of a plan generation system. Research Report CBM-TR-128 Rutgers University.

[13] Vignard P.; CRIQUET : un outil de base pour construire des systèmes experts. Rapport de Recherche INRIA No 316 Juillet 1984.

[14] Wilensky R., Arens Y., Chin D.; Talking to Unix in english. Communications of the ACM, 1984, 27, 6, 574-593.

[15] Wilkins D.E.; Domain-independant planning: Representation and Plan Generation. Artificial Intelligence 22 (1984) 269-301

Imprimé en France

par

l'Institut National de Recherche en Informatique et en Automatique